Team 1 Design Document: WeChef

Team Members:
Sean Chew (schewshu@purdue.edu)
Zuyuan Fan (fan164@purdue.edu)
Yuting Guo (guo312@purdue.edu)
Qi Meng (meng46@purdue.edu)
Ling Zhang (zhan2275@purdue.edu)

Table of Contents

Purpose

Have you ever found yourself standing in the kitchen with a handful of ingredients that you have no idea what to do with? Oftentimes, just finding the correct recipes can be a hassle. Our team is envisioning a convenient and user-friendly mobile app called WeChef as the solution to this problem.

The mobile app will provide individuals with an easy way to discover new recipes as well as find recipes based on the current available ingredients at hand. In addition, WeChef will allow individuals to live stream their cooking sessions for the general public to view and learn from them. They will also be able to upload photos/videos for individuals who prefer to learn at their own pace.

What makes us stand out from our competitors, such as Tasty, is that our app is user-based, whereby anyone can upload their recipes. Tasty, on the other hand, is fully controlled by the professional editors, and users can only view the recipes posted by them.

We have two main purpose that we aim to achieve while developing our app. The first is that our app aims to provide a learning platform for anyone who aspires to improve their cooking skills. The second is to grow a fun and friendly community for cooking enthusiasts and allow them to expand their social circles with individuals who share similar interests.

- Functional Requirements

1. As a user, I want to create an account
2. As a user, I want to sign in to the application
3. As a user, I want to sign out of the application
4. As a user, I want to set a profile name
5. As a user, I want to set a profile picture
6. As a user, I want to update my profile name

7. As a user, I want to update my profile picture
8. As a user, I want to search for a particular recipe from the search bar using the recipe name
9. As a user, I want to search for a recipe using particular ingredients
10. As a user, I want to see ingredients and the amount used in the recipe
11. As a user, I want to see step-by-step details of the recipes
12. As a user, I want to see pictures of the dish posted by the author of the recipe
13. As a user, I want to watch cooking videos posted by the author of the recipe
14. As a user, I want to see pictures of dishes cooked by other users who used the recipe
15. As a user, I want to see comments and questions posted by other users who used the same recipe
16. As a user, I want to add ingredients of a recipe to my shopping list
17. As a developer, I want to provide the users unit conversion function
18. [If time permits] As a user, I want to view my shopping list by recipe
19. [If time permits] As a user, I want to combine recipe ingredients in shopping list
20. [If time permits] As a user, I want to remove all ingredients from my shopping list
21. [If time permits] As a user, I want to remove ingredients in a recipe from my shopping list
22. [If time permits] As a user, I want to see the homepage of the author of a particular dish
23. [If time permits] As a user, I want to add a particular recipe to favorites
24. [If time permits] As a user, I want to like a particular recipe
25. As a user, I want to post detailed steps of my own recipes
26. As a user, I want to post pictures of my own recipes
27. [If time permits] As a user, I want to post videos of my own recipes
28. As a user, I want to ask questions about a particular recipe
29. As a user, I want to post comments under the recipes that I have used
30. As a user, I want to post pictures of my version of the dishes under the recipes that I have used

31. As a user, I want to rate subjective quality of the recipes that I have used
32. As a user, I want to rate subjective difficulty of the recipes that I have used
33. As a user, I want to edit the recipe that I have posted
34. [If time permits] As a user, I want to follow the users whom I am interested in

- Non-Functional Requirements

- Architecture
  We've decided to develop this application on iOS since it is one of the most popular mobile operating systems, and so we could utilize this large platform to promote this application.

  We will separate our product to data layer, logic layer, and presentation layer. Data layer consists of our database and object storage. Logic layer connects and communicate between data layer and presentation layer to respond to requests from our frontend client. Client application interface serves as the presentation layer of our product; it displays information to end users and allow them to use the core functions of the application.

  By separating the product into the layers, we eliminate dependency among function components and gain more flexibility on choices of framework and tools. This also ensures scalability of server, database, and object storage in an on-demand manner.

- Framework and Tools
  We will construct our API in Node.js environment with Express framework to take care of basic HTTP infrastructures, like routing and handling requests. The API route can be triggered by the client application and perform backend logic and database operation according to the requests. The API will be hosted on Heroku for its convenience in deployment and

management. We will store information of users and recipes in MongoDB served on mLab added on Heroku. Photos will be stored with another Heroku add on, .

For the client side, we will be using React Native as a front-end development framework. Facebook will be used for user authentication proposes to make the sign up/login process more convenient and secure.

- Security
  As our application will be built on matured platforms like Node JS Express, MongoDB, etc, our application will be equipped with security features that are provided by these services. For example, MongoDB provides various security features that include authentication, access control, encryption, etc. In addition, we will extensively use APIs, such as Facebook Login. Facebook Login provides features, such as access tokens and permissions, to make it safe and secure for applications to use.

- Usability
  The success of our app is ultimately determined by our users. There, usability and user experience is of utmost importance to us and will be a driven factor of our UI/UX design process. We try to make the interface as user-friendly as possible, eliminating as much hassle as we can along the way. Starting from the login and sign-up process, we only allow for third-party authentication login (such as Facebook login) to free our users from memorizing yet another username and password. This also eliminates the initial complicated sign-up process and allows users to dive right in and enjoy the exciting services right away. Once logged in, the user-friendly interface allows our users to quickly familiarize themselves with the app to utilize the full-range of functionalities. In addition, since our application will involve extensive use of pictures and possibly videos, we want to provide a smooth picture and video loading process by integrating lazy loading feature.

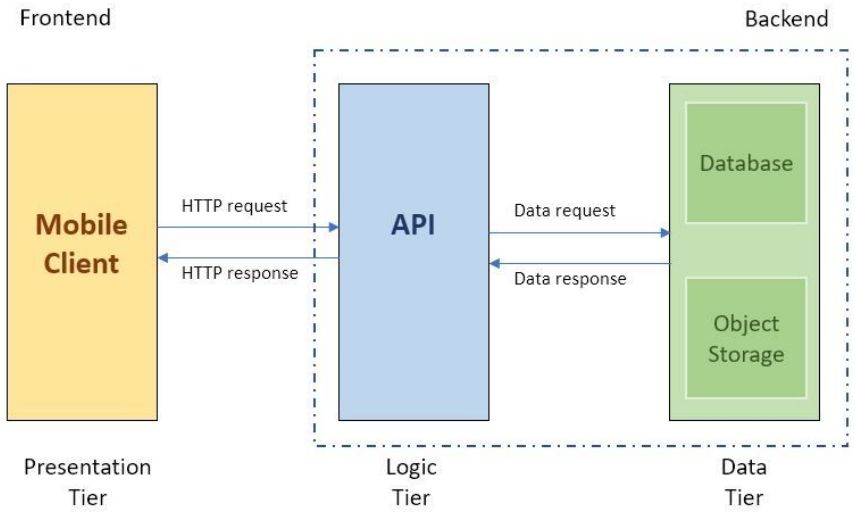
- Scalability

Given that our original thought is that our product will be used by our group members and some of our friends, there may not be many problems with scalability. If our product were to be released to a wider group of users, we can easily upgrade Heroku Dynos and increase the Cloudinary and mLab storage in order to sustain a relatively high speed and keep the user data.

Design Outline

- Design Decisions and Components

Our application will use a 3-tier client-server architecture. The presentation tier is the mobile client app, directly displaying information to users and sending user-defined requests to the logic tier in the form of HTTP requests. The RESTful API servers as the logic tier of the application, which handles incoming requests from clients, communicate with data tier, and feedback to the presentation tier with suitable responses. The data tier and logic tier together consist of the backend of the application.



- Client-Server(API) Interactions

• Client sends request to the API. For example, client requests to get detailed content of a recipe.
• Client receives responses from the API. For example, API responses to the client with detailed content of the requested recipe.
• Retrieved information will be parsed and then displayed to the user.

- Server(API)-Database Interactions

• API issues database operation to read desired data needed to process the received requests. For example, API finds a recipe document in the database by recipe ID.
• API receives data from the database. For example, API receives the recipe document with the requested recipe ID from the database.
• API issues database operation to write data specified by the received requests. For example, API update the recipe name of a recipe document in the database by recipe ID.
• API receives result of the write operation from the database. For example, API receives the a success for updating the name of the recipe document with the requested recipe ID from the database.
• API issues database operation to delete a document specified by the received requests. For example, API delete the recipe name of a recipe document in the database by recipe ID.
• API receives result of the write operation from the database. For example, API receives the a failure for deleting the recipe document as there is no recipe with the requested recipe ID in the database.
• Retrieved document or result of operation will be processed and then send to client.

- High Level Overview

The application consists of 4 components described below:

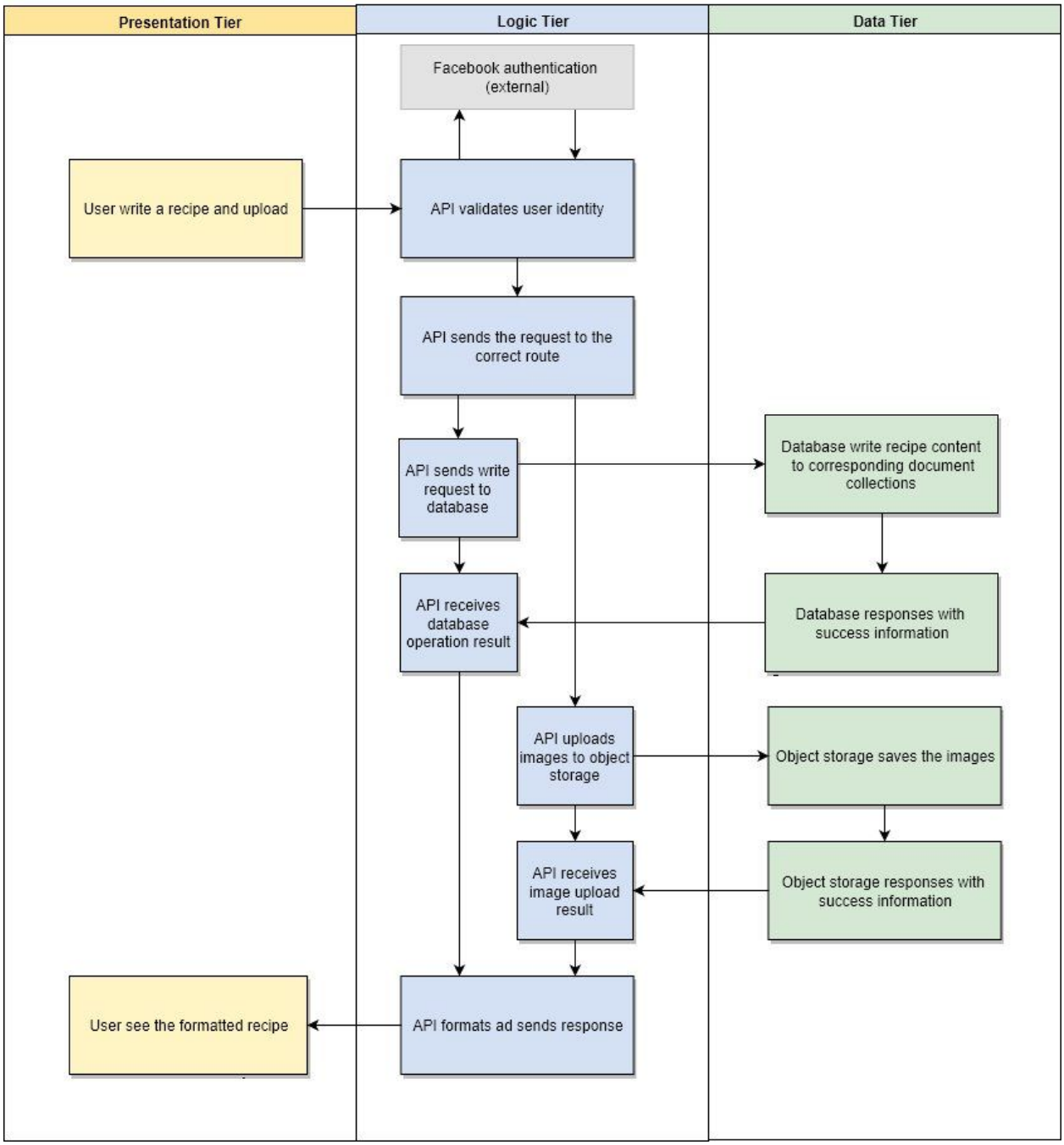
1. Client
    a. Resides in iphone
    b. Processes and format user inputs
    c. Sends HTTP requests to the API
    d. Receives HTTP responses from the API
    e. Processes, format, and display information received
2. Heroku Express+Node.js API
    a. Receives HTTP requests from the client
    b. Authenticates the user identity according to information in requests
    c. Decodes and process the incoming HTTP requests
    d. Sends requests to database and/or data storage
    e. Processes received responses from database and/or data storage
    f. Sends HTTP responses to the client
3. MongoDB Database
    a. Stores user information, including:
        i. User profile and location of profile picture
        ii. List of recipe IDs of the recipes which the user created
        iii. List of recipe IDs of the recipes which the user favorites
        iv. List of recipe IDs of the recipes which the user added to shopping list
    b. Stores recipe content, including:
        i. User ID of the author of the recipe
        ii. Recipe title
        iii. Location of recipe images and/or video
        iv. List of recipe ingredients and corresponding amounts
        v. Cooking steps
        vi. Rating of the recipe
        vii. Difficulty level of the recipe
    c. Stores Q&As under recipes, including:
        i. Recipe ID of the recipe to which the Q&A belongs to

ii. Content of the question
iii. User ID of the user who asked the question
iv. Content of the answer provided by the recipe author

d. Stores reviews under recipes, including:
i. Recipe ID of the recipe to which the comment belongs to
ii. Text content of the review
iii. Location of the image in the review

e. Receives and process request from the API
f. Perform database operations specified by the request from API
g. Send responses to the API

4. Cloudinary Object Storage
a. Stores user profile pictures
b. Stores recipe images/videos


Presentation Tier
Logic Tier
Data Tier
Facebook authentication (external)
User write a recipe and upload
API validates user identity
API sends the request to the correct route
API sends write request to database
Database write recipe content to corresponding document collections
API receives database operation result
Database responses with success information
API uploads images to object storage
Object storage saves the images
API receives image upload result
Object storage responses with success information
User see the formatted recipe
API formats ad sends response

Design Issues

- Functional Issues

➔ Do users have to create unique credentials to login to WeChef?

- Option 1: Users set up username/password.
- Option 2: Users use third party login.

Reason: We choose to let users login using third-party login to avoid memorizing another username and password. This also eliminates the initial complicated sign-up process and allows users to dive right in and enjoy the exciting services right away.

➔ Should users be able to add comments and rate all recipes?

- Option 1: Users are able to comment under any recipe.
- Option 2: Users are allowed only comment under the recipe they have used.

Reason: Considering that comments and ratings are fair if and only if a user has used the recipe, we will only allow users to comment the recipe which they have used. If a user upload a photo of “my version” dish, he or she will be identified to have used the recipe.

➔ When searching for recipes, what will be included in the results?

- Option 1: The results only show recipes whose titles include the searching keywords.
- Option 2: The results show recipes whose titles include the searching keywords, as well as recipes whose ingredients include the searching keywords.

Reason: Since users may not know exactly what they want to cook, it would be more user-friendly to also show users the recipes that have the ingredients specified by the user.

➔ What should be displayed on the homepage?

- Option 1: The first ten or less latest recipes are displayed.
- Option 2: The recipes added to the user’s favourite list are displayed.

Reason: If the recipes being added to favourite are displayed in the homepage, the user will always see the same set of recipes. Instead, if the latest recipes are displayed, the user is able to see the trending recipes and almost different recipe feeds every time.

➔ Should we allow all users to upload new recipes onto the app?

- Option 1: All users are able to upload new recipes.
- Option 2: Only users who are certified as professionals or top-rated users are able to upload new recipes.

Reason: As our app aims to provide a platform for all cooking enthusiasts to share their passion, we will allow all users, such as regular homecooks, to be able to upload new recipes onto the app. This is also the main feature that differentiates us from our competitors.

- Non-Functional Issues

➔ What platform will the app be on?

- Option 1: Mobile
- Option 2: Web

Reason: Considering that most users use mobiles phones more often than laptops in their daily lives, it is more useful if we host the app on mobile. It would also be more convenient to hold a phone while cooking rather than holding a laptop.

➔ What iOS development language and framework will we use for the app?

- Option 1: Swift
- Option 2: JavaScript and React Native

Reason: Considering the popularity and compatibility of React Native and JavaScript, it is more beneficial if we develop our app using these two language and framework.

➔ What backend language and frameworks will be used?

- Option 1: JavaScript, Node.js, and Express
- Option 2: Python and Flask

Reason: Our frontend is written in JavaScript, writing the backend in JavaScript makes the maintenance of code base easier. Although Express involves asynchronous functions, we have members who have experience in Express; therefore, we selected JavaScript, Node.js, and Express.

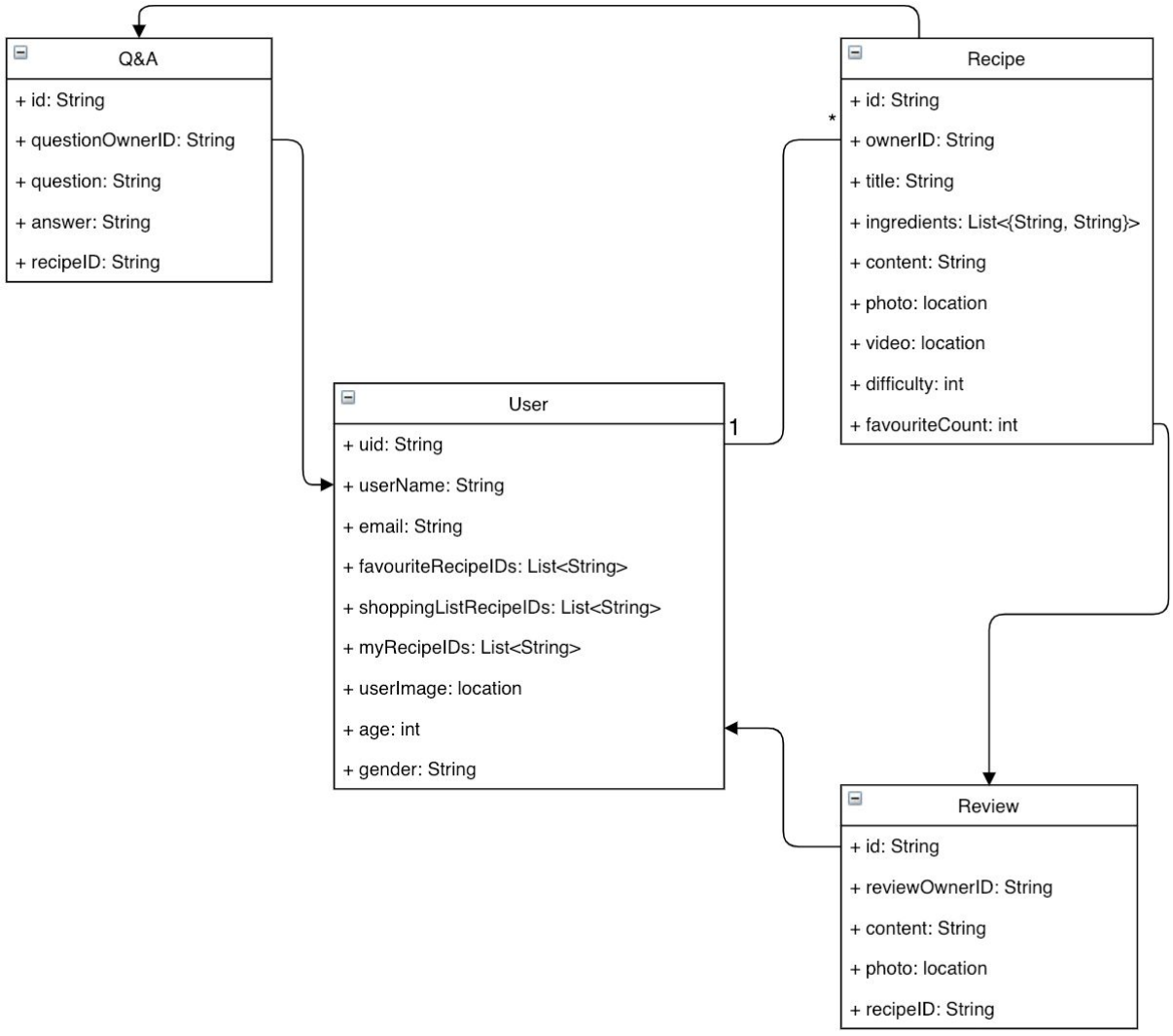
➔ What type of database will we use for the app?

- ◆ Option 1: MongoDB
- ◆ Option 2: DynamoDB

Reason: Although MongoDB and DynamoDB are NoSQL databases, DynamoDB is more closely coupled with other AWS services, while MongoDB is adaptable anywhere. As we choose to use Express and Node.js to build our API and deploy on Heroku, MongoDB can be easily added to Heroku without having to configure and host the database separately. Therefore, we choose to use MongoDB as our database.

Design Details

- Class Diagram



- Description of Classes

➔ **User:**

- Represents the current user
- Created when a user first login using Facebook
- Contains email address and auto generated user ID for developers' use
- Contains all the detailed information of a user, including username, profile picture, age and gender
- Contains a user's favorite recipe album, including all the recipes that the user saved
- Contains a user's shopping list, including all the recipes that user add to chart

➔ **Recipe:**

- Created when a user upload his/her recipe
- Contains unique ID for developers' use
- Contains all the information regarding the recipe, including owner id of the recipe, title, ingredients, instructions, photos, and difficulty
- Contains the video tutorial for the recipe
- Contains the number of users who added the recipe to their favourites

➔ **Q&A:**

- Created when a user first ask a question on a recipe
- Contains unique ID for developers' use
- Contains the ID of the recipe that the Q&A belongs to
- Contains the ID of the user who posted the question
- Contains question String and answer String

➔ **Review:**

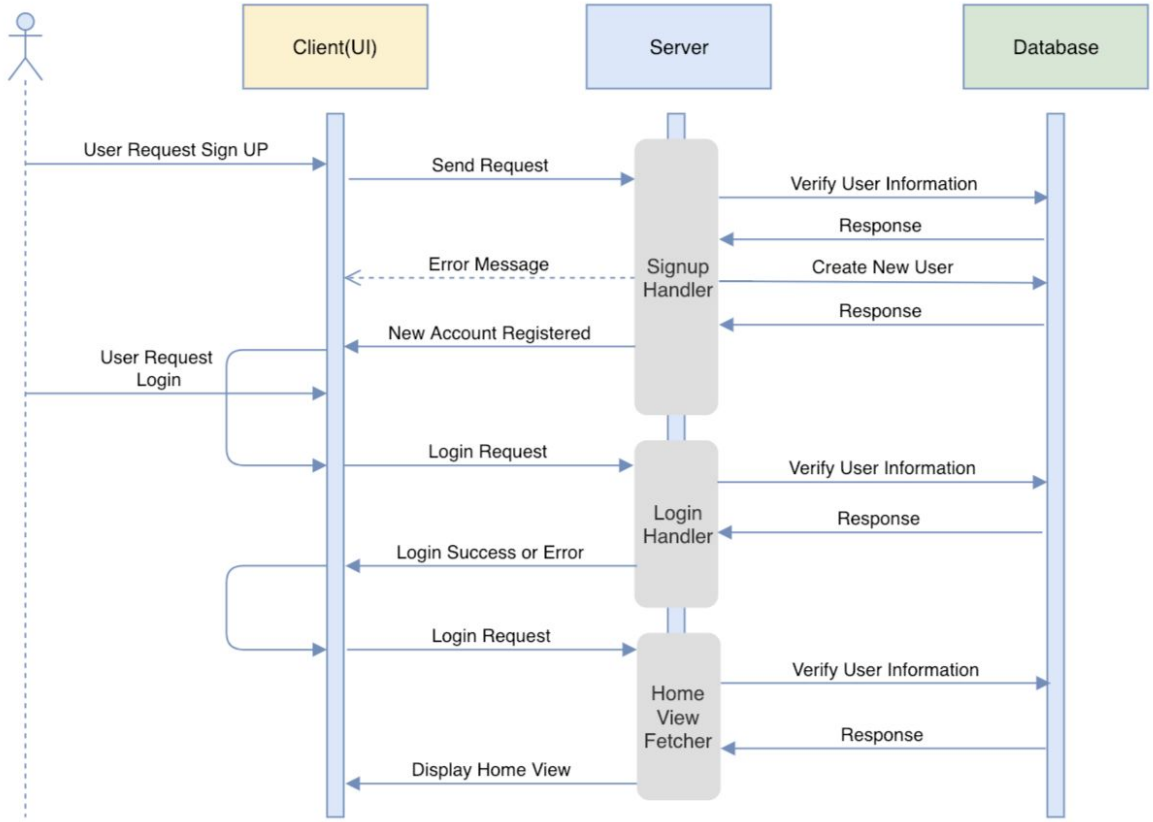
- Created when a user first post a review on a recipe
- Contains unique ID for developers' use
- Contains the ID of the recipe that the review belongs to
- Contains the ID of the user who posted the review
- Contains a String content of the review
- Contains a photo of a dish cooked by the author of the review

- Interactions Between the Classes

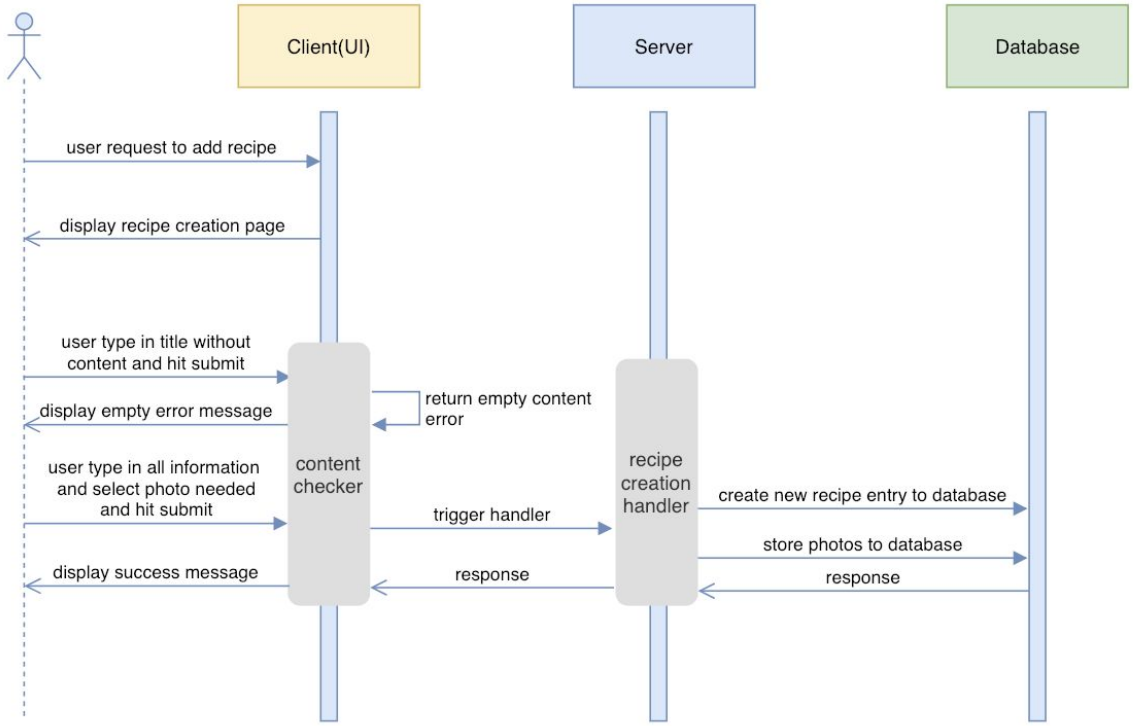
The User class interacts with the Recipe class through foreign key ownerID. When a user creates a new recipe, a new recipe object is created and the user id is used as the ownerID of the new recipe. The User class also interacts with the Q&A class and the Review class via foreign keys questionOwnerID and reviewOwnerID, respectively. When a user leave a question or write a review, a new Q&A or review object is created with the user id as the owner id. In addition, the Recipe class interacts with the Q&A class and the Review class using recipeID in them. When new Q&A or questions are created under a recipe, the recipe id will be added to them.

- **Sequence Diagrams**

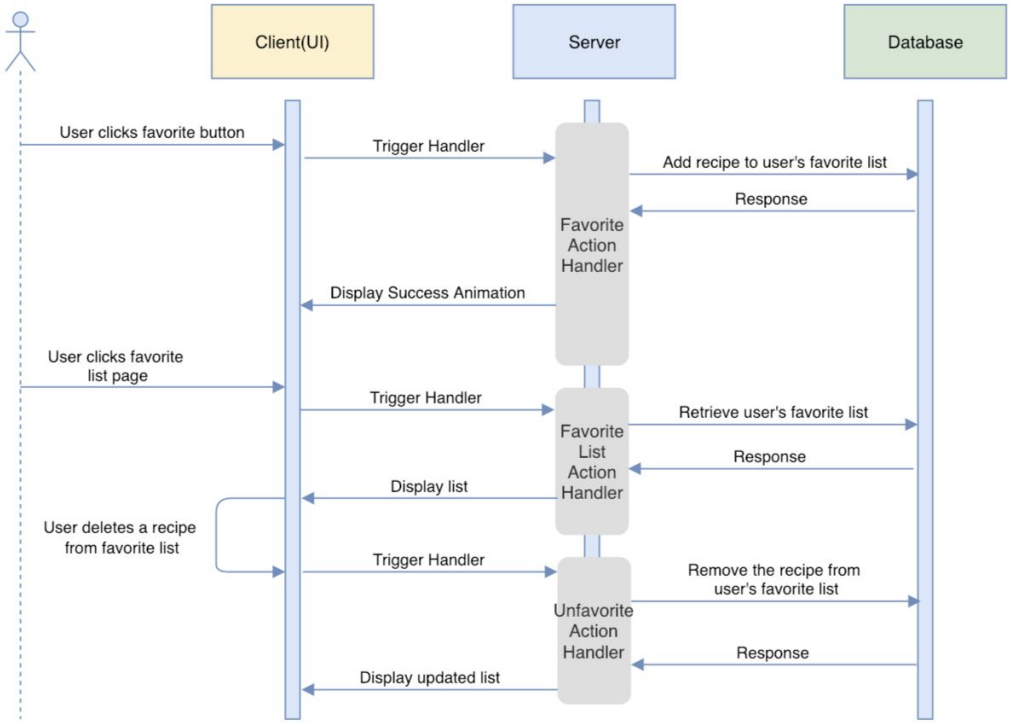
➔ Sequence of events after the user starts the application



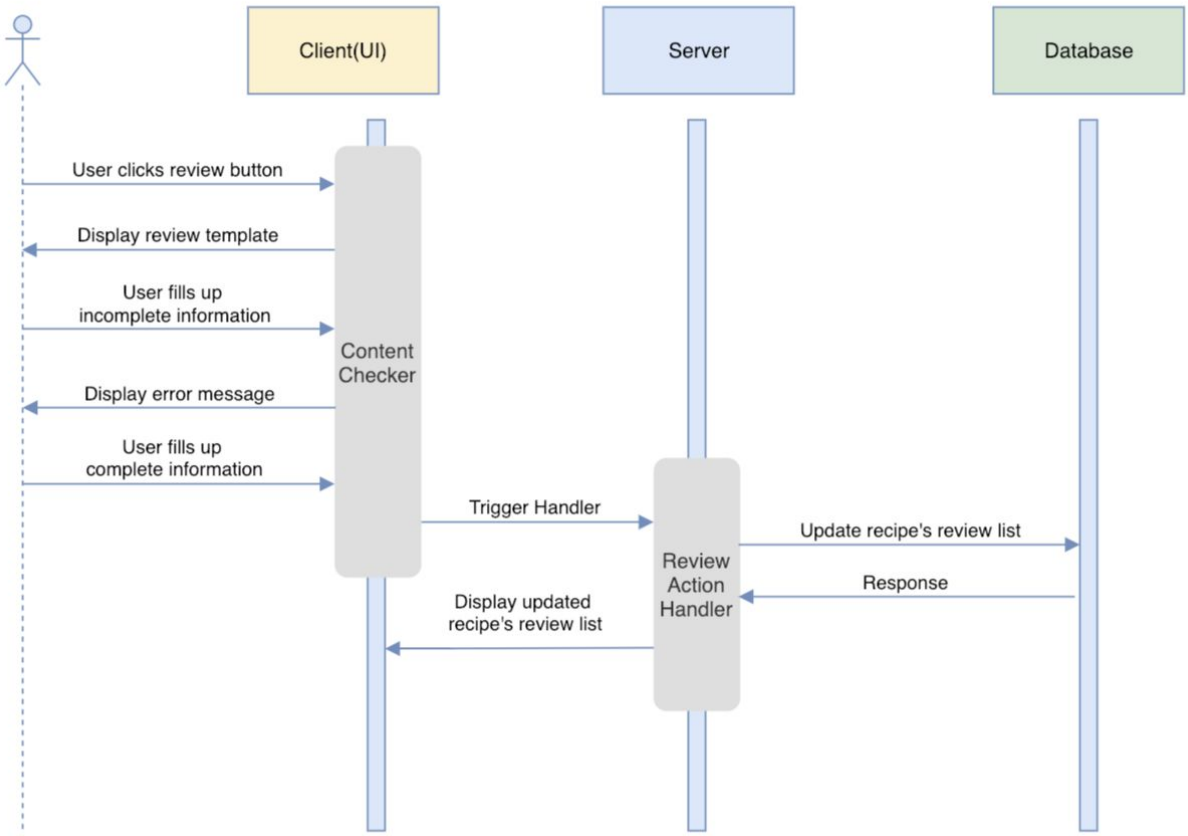
➔ Sequence of events when user creates a new recipe



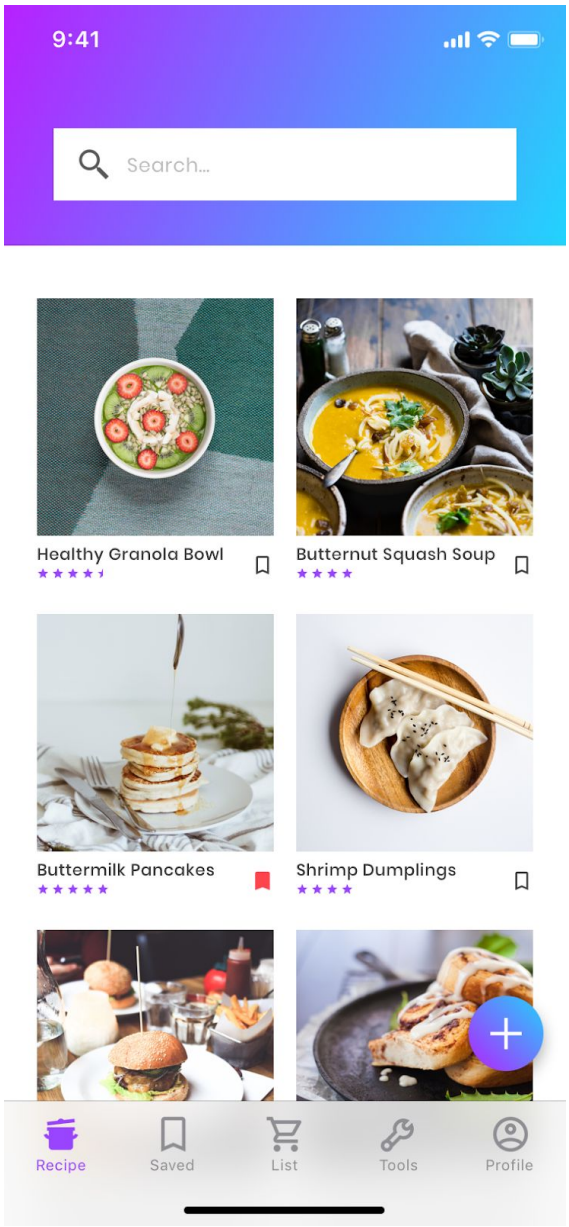
➔ Sequence of events after the user adds a recipe to favorite

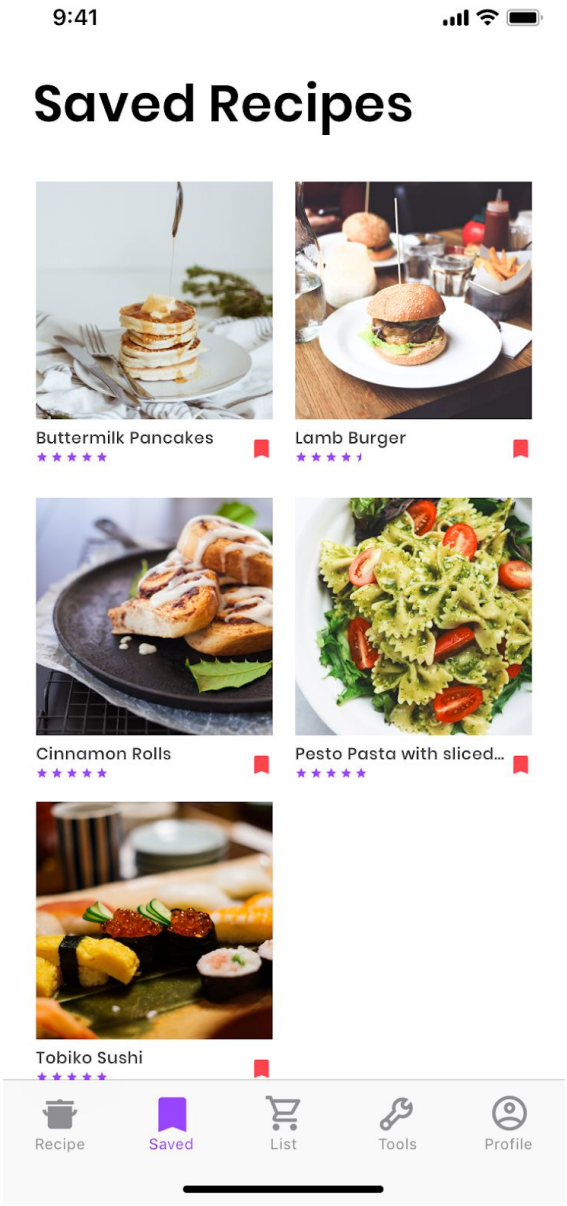


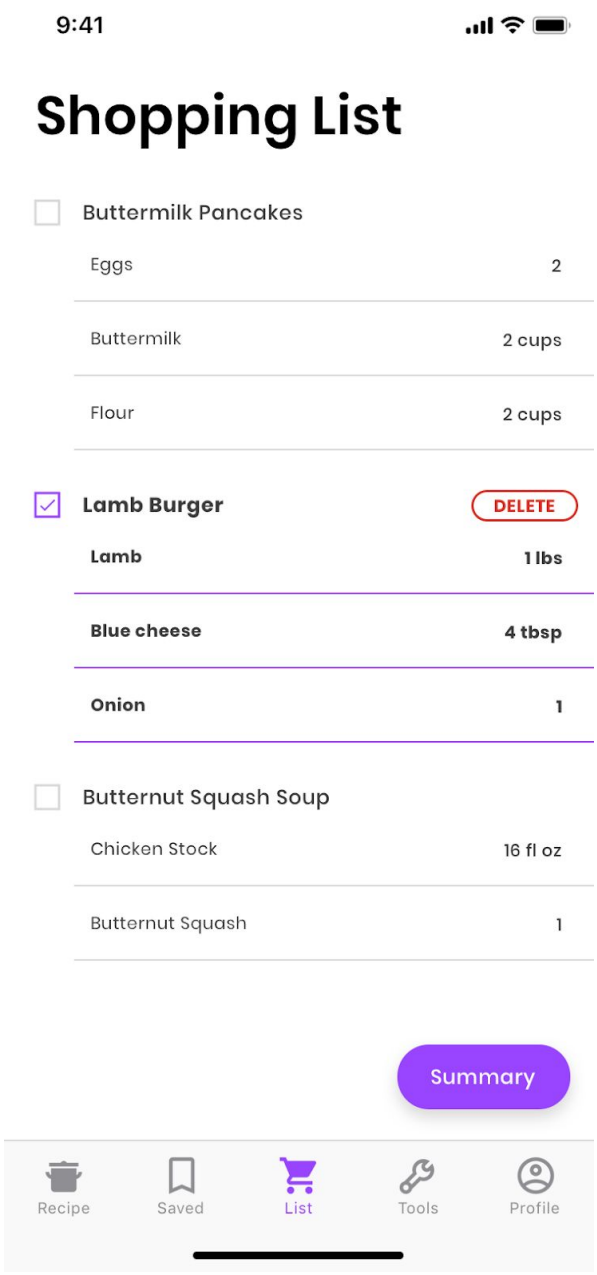
➔ Sequence of events after the user adds a review to a recipe

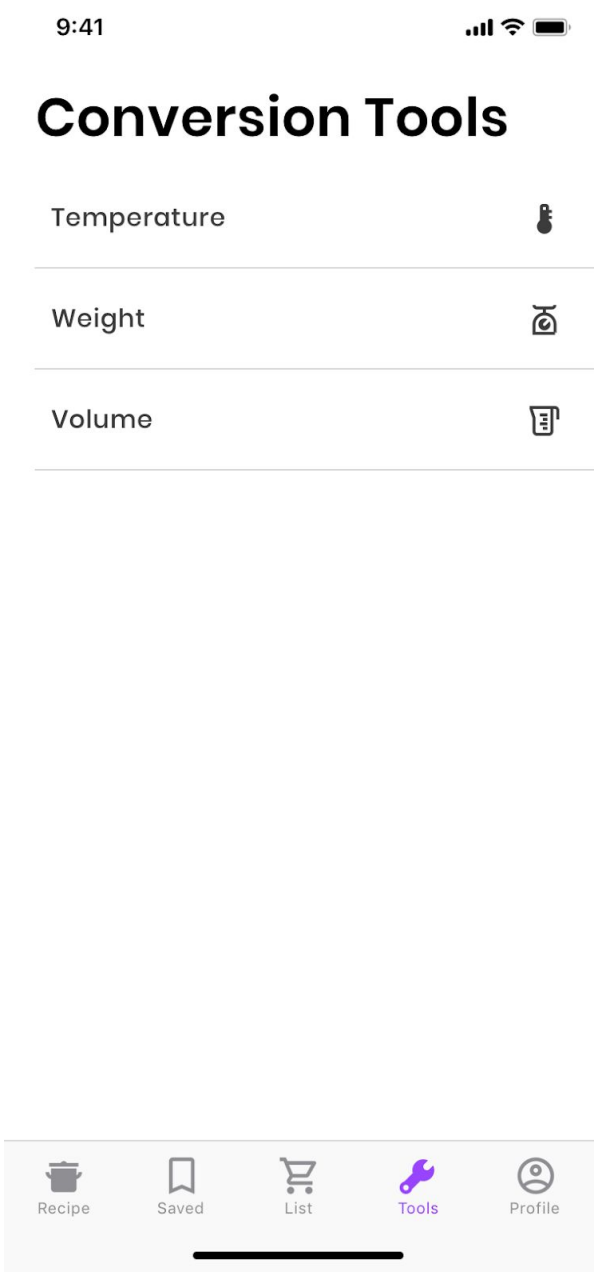


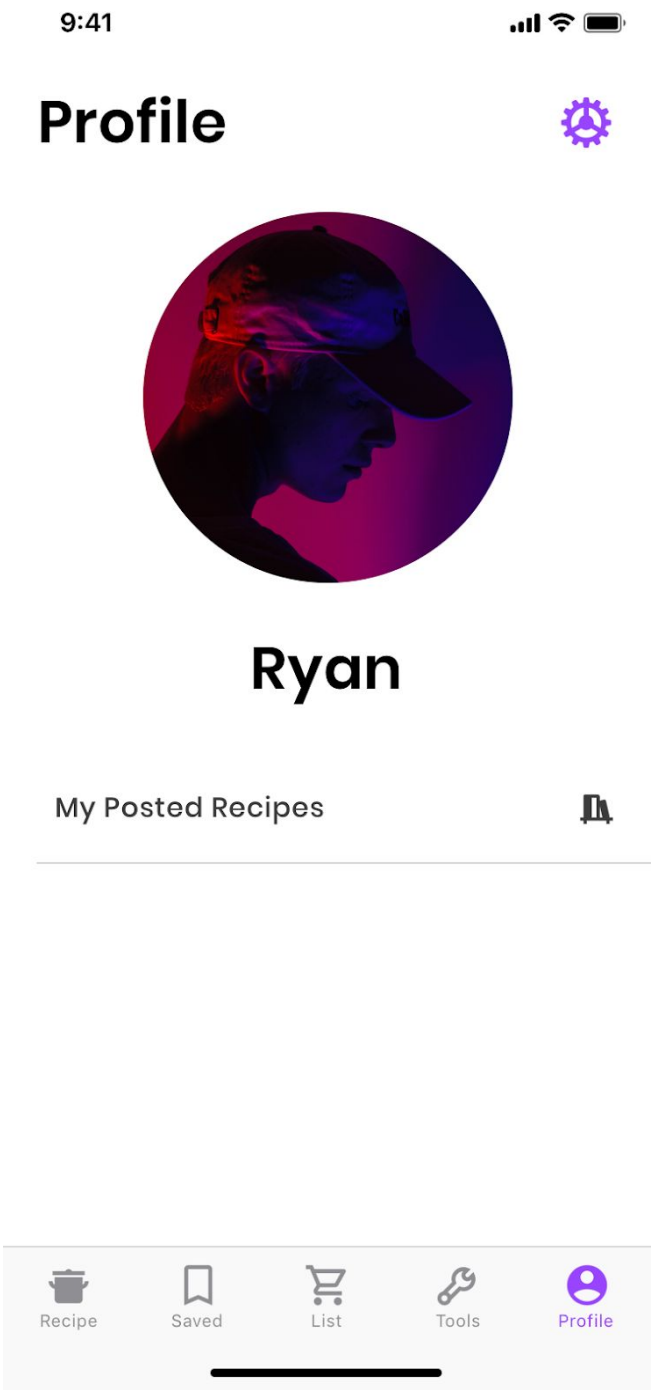
- **UI Mockups**


9:41
Search...
Healthy Granola Bowl
Butternut Squash Soup
Buttermilk Pancakes
Shrimp Dumplings
Recipe
Saved
List
Tools
Profile


9:41
Saved Recipes
Buttermilk Pancakes
Lamb Burger
Cinnamon Rolls
Pesto Pasta with sliced...
Tobiko Sushi
Recipe
Saved
List
Tools
Profile


9:41
Shopping List
Buttermilk Pancakes
Eggs 2
Buttermilk 2 cups
Flour 2 cups
Lamb Burger
DELETE
Lamb 1 lbs
Blue cheese 4 tbsp
Onion 1
Butternut Squash Soup
Chicken Stock 16 fl oz
Butternut Squash 1
Summary
Recipe
Saved
List
Tools
Profile


9:41
Conversion Tools
Temperature
Weight
Volume
Recipe
Saved
List
Tools
Profile


9:41
Profile
Ryan
My Posted Recipes
Recipe
Saved
List
Tools
Profile